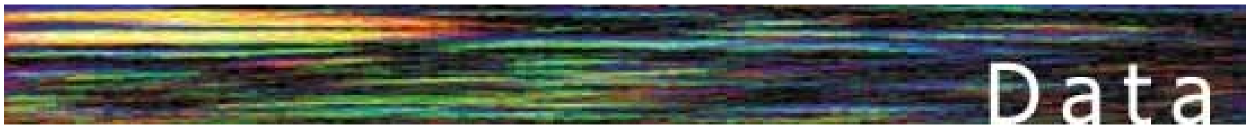


Large File Performance

By Dale Fredrick

With the introduction of 64-bit MKEYED files in Revision 3.0 of PRO/5� and Visual PRO/5�, the ability to generate extremely large data files is now a reality. Along with the ability to generate very large MKEYED files come questions about severe degradation in file operation performance as records are added to the file. We did not believe that there was a point at which users of large MKEYED files would "hit the wall" and suffer an extreme loss of performance, but we didn't have any testing or data to back it up. Based on some original testing done by the BASIS Quality Assurance (QA) group, we set out to gather some real data on large MKEYED file performance. The simple benchmark and the results of that work are presented in this article.

The Benchmark

The initial composition of the benchmark can be attributed to a test written originally by the BASIS QA group. The original intent, which has been maintained, was to generate a 64-bit MKEYED file in batch mode. The file would be grown in roughly 500-MB chunks and would consist of nine 30-byte keys, with a total record length of 998 bytes. By experience, the test would use one of the more time-consuming operations that occurs with an MKEYED file, that of inserting a new record into a file. The BBX� program would be run 14 times and insert 305,000 records in each batch run.

The records are inserted in random order. The first run would insert records containing keys from 1 to 305,000; the second run would insert records containing keys from 305,001 to 610,000, etc. Each of the nine keys for a given record has a different random value.

I took over the test when the QA group was getting results that were all over the map and rather confusing. My original goal was to determine why the QA tests had large fluctuations in the results from one execution of the benchmark to another. We did not believe that the MKEYED driver or PRO/5 was at fault. In the end, we found that our laboratory configuration, which relies on NIS to provide password information and other UNIX database functionality, was partially at fault. We reconfigured for a local user, avoiding both the use of NIS and NFS during the tests, disallowing the use of the machine by all other users and implementing a local license manager. After this reconfiguration, multiple executions of the benchmark were generating consistent results from one execution to another.

Host Configuration

The benchmark has been repeatedly executed on three separate hosts in the BASIS Engineering/QA laboratory:

Host Type	Operating System	Host Memory	Disk Where Test File Resides
HP Model	HP-UX 10.20	32 MB	Seagate ST19171N (9 GB)

712/60			
Motorola MT604-100	AIX 4.3.2.0	64 MB	Seagate ST19171N (9 GB)
IBM RS/6000 43P Model 260	AIX 4.3.2.0	512 MB	SSA Multi-Initiator/RAID EL adapter with SSA fast-write cache option. Three striped UltraStar 9LP (9 GB) disk drives in a SSA multi-storage tower.

Each of the hosts has been configured with a minimum of three local file systems. The file systems are built in a configuration in which each utilizes a separate disk drive or disk drives:

- root
  which contains the operating system, /usr and other standard UNIX operating system directory structures. Under AIX, /usr, /var and other operating system-related structures are different file systems than that of root. In our configuration, however, they reside on the same physical disk drive.

- local Home
  The PRO/5 3.0 executable, the BBX program and any called programs, shell scripts and the resulting log files reside on this file system.

- test area
  contained the file being tested.

The file system and directory structure layout is done in this manner so that I/O traffic to the test file, which resides on the test area file system, can utilize separate I/O adapters and/or controllers. This layout also allows the data to/from the file under test to be the only active file on a disk or striped disk set. An additional benefit of this configuration is that statistics about I/O activity at various levels of the operating system and the physical disk drive can also be restricted to the disk containing the file under test.

The bourne shell script, which drove the test, was submitted to the system for execution by use of the at(1) command. On the IBM RS/6000 43P Model 260, the root and home disks were on separate interface buses and adapters than that of the 9-GB striped disks.

The Results

The results of the test for each of the three nodes is shown in Figures 1, 2 and 3. The graphs are plotted with time on the Y axis and the benchmark run number on the X axis. The UNIX time(1) command was used to execute each run of PRO/5 and to provide the timing results from the operating systems process resource usage statistics. After each run of PRO/5 finished executing, the time command reported (in seconds):

- real = the total real time elapsed
- user = the time used to execute the PRO/5 process in user mode
- system = the time consumed by system overhead in the process
- u+s = the sum of the user and system times

Figure 1. Performance Results for PRO/5 3.0 on HP Hardware.

HP Model 712/60

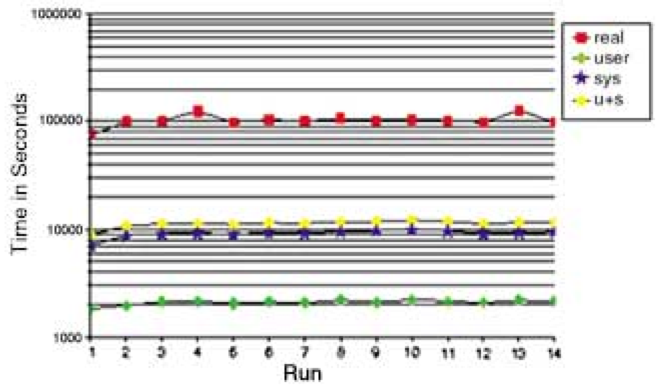


Figure 2. Performance Results for PRO/5 3.0 on IBM Hardware.

IBM RS/6000 43P Model 260

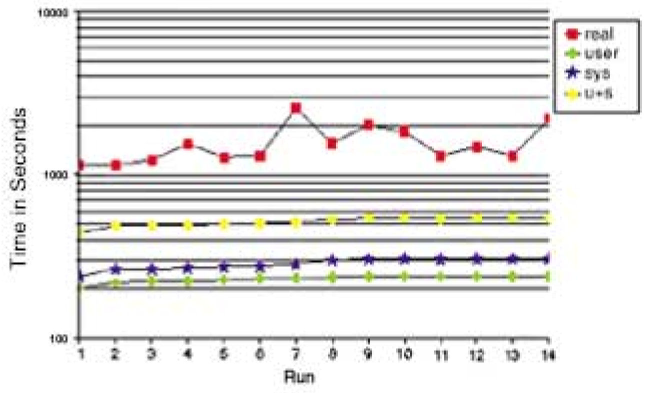
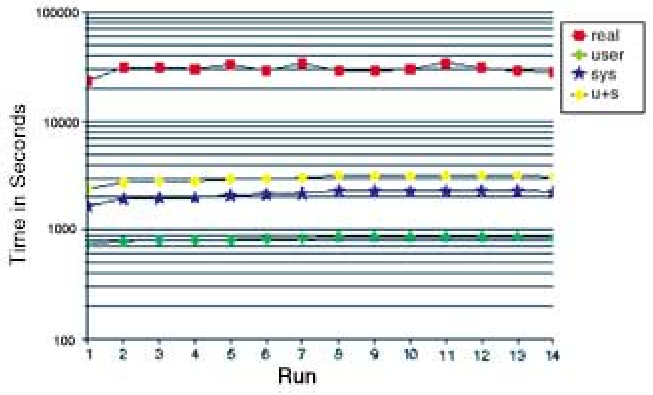


Figure 3. Performance Results for PRO/5 3.0 on Motorola Hardware.

Motorola MT604-100



After the first two runs, the file had grown to approximately 1 gigabyte in size and contained 610,000 records. The time taken for the addition of records to the file by the following batch insertion runs stays reasonably consistent from Run 3 to Run 14. After Run 14 is completed, the file has achieved a total size of approximately 7.2 gigabytes and contains 4,270,000 records.

As can be seen from the graphs, once the file with the defined configuration has reached a size of approximately 1 gigabyte, the time it takes to add additional records to the file continues to increase slightly in the user and system times. This is to be expected. However, it does not increase suddenly or even dramatically with the following 12 runs on any of the hosts where the test was run.

You may have noticed that the real time values in all the graphs increase and decrease variably from run to run. PRO/5 does not do synchronous writes to the actual disk but uses the standard UNIX read and write calls to read and write the file header, key nodes and records to the file. Under modern UNIX operating systems, files are buffered in some sort of disk cache. A few versions of UNIX still use the older buffer cache and have some kind of sync daemon to flush modified blocks back to disk.

Operating systems like AIX today still have a disk I/O cache, but it has been merged with the Virtual Memory Management subsystem. Under AIX, normal access through the Journaled File System (JFS) is managed by the Virtual Memory Manager (VMM) and does not use the traditional method for caching the data blocks. As blocks of a file are requested by a read operation, if they are not already present in the VMM disk cache, a page fault occurs and they are paged in from disk. When blocks are modified by a write operation, the blocks in the VMM disk cache are changed. The movement of that data from the VMM disk cache to the physical disk takes place outside of the actual process by a delayed write mechanism within the VMM. PRO/5's use of the UNIX write system call utilizes writes that are asynchronous. The end result in our test shows that FIFO I/O queues of several megabytes build up and consume considerable time by the VMM while being moved to disk, causing the fluctuations in the real time from run to run.

Conclusion

The tests clearly show that the performance of the MKEYED file driver in the PRO/5 file system scales extremely well into multiple gigabyte file sizes. We realize that we have just begun to scratch the surface when it comes to file performance testing. We will continue a program of file performance testing both with PRO/5 and BBjTM as part of our efforts to provide our customers with a modern high-performance, high-reliability file system.

References

For more information about performance and tuning for two of the platforms we tested, I recommend the following books. They further define I/O workload measurement and tuning.

Sauers, Peter S. and Weygant, Robert F., *HP-UX Tuning and Performance: Concepts, Tools, and Methods, First Edition*, Prentice Hall, July 9,1999.

Waters, Frank, *AIX Performance Tuning Guide, First Edition*, Prentice Hall, September 28, 1995.